Syracuse University
L.C. Smith College of Engineering and Computer Science

CIS 556 – Principles of Programming II
Spring 2003

Catalog Description
Functional programming, modules, data encapsulation, types and polymorphism, programming correctness, lambda-calculus, beta-reduction, evaluation strategies, basic implementation techniques.

Instructor Information
Prof. Susan Older  (sueo@ecs.syr.edu)
Office: CST 3-179, x4679
Office hours: Tues & Fri 1:30-3pm, or by appointment

Course Web Address
http://www.cis.syr.edu/~sueo/cis556

Textbook

Additional References
Other notes will be distributed during the semester as necessary.

Prerequisites
In this course, we will be building on foundations established in CIS 521 and CIS 555.

In CIS 521, you covered basic set theory, function, predicate logic, and induction. More importantly, you should have gotten experience thinking with mathematical rigor. In this course, we will make use of these experiences in several ways. First, we’ll be using the functional-programming language Haskell: much of Haskell’s syntax has been designed to look like mathematical notation in order to make explicit the underlying mathematical ideas. For example, Haskell lists can be defined in a notation that looks like set comprehension. Second, an important feature of programs written in functional languages is that they are typically much easier to reason about and prove correct than those written in imperative languages. We will be using logic and induction to help reason about the behavior of the programs we write. Finally, we will make significant use of inductively defined data types and data structures: understanding induction is essential for understanding the validity of these beasts and for correctly implementing algorithms that operate on them.

In CIS 555, you saw many data structures—including stacks, queues, lists, and trees—as well as algorithms that operate on them. In this course, we will frequently use trees as internal representations of program code (these representations are known as “abstract syntax”) that we then manipulate in various ways. We will make use of stacks (or related structures) to implement concepts such as program environments or stack frames.
Course Objectives
The primary objective of this course is to provide you significant experience with functional programming, which tends to enforce a more abstract and mathematical view of programming than other paradigms do. It supports a view of component-based design that is especially well-suited for problem solving and for logical analysis. Consequently, you will begin to think about programming in a different, more systematic and rigorous way, even as you return to imperative and object-oriented programming.

A second objective is to provide you with significant experience with typeful programming. Many program bugs are really due to mismatches between types: a function or procedure is expecting one sort of input but receives something else. These sort of mismatches typically reflect logical errors in thinking. A language that has a static type system (e.g., Haskell or Java) enforces the correct use of types at compile or interpretation time: as a result, you can catch many logical errors before a program is run. Experience with typeful programming helps you develop more precision of thought, thereby becoming a better programmer. Again, this ability stays with you, even when you return to other styles of programming.

Course Outcomes
After completion of the course, students should be able to:

- Read, write, and understand programs written in the functional paradigm (particularly, in the language Haskell)
- Use induction and other mathematical apparati to prove properties about Haskell programs
- Deduce the result of applying type-checking and type-inference algorithms
- Implement in Haskell nontrivial applications, such as type-inference algorithms, rule-based systems, and language parsers or evaluators

Outcome Measurement
Your final grade will be based on a variety of activities:

- Homework assignments (40% of final grade)
  Homeworaks are intended to keep you up-to-date with the material. As such, there will be a homework assignment approximately every week. Occasionally, students may be asked to explain their homework to me or to the class: in such cases, the homework grade will be based on the results of this explanation.

- Quizzes (35% of final grade)
  There will be a quiz every two or three weeks: their purpose is to let you demonstrate your understanding of the fundamental concepts. I will drop the lowest quiz grade at the end of the semester.

- Programming project (25% of final grade)
  The purpose of the programming project is to let you apply Haskell to a nontrivial application.

You should also be aware of the following submission practice and policy:

Occasionally, a homework assignment may indicate that you can work in pairs; otherwise, you must work alone. If you work with someone, you should submit a single
solution: each person’s name must appear clearly on the first page. All members of a group will receive the same grade (even if only one student is asked to explain a homework); groups of more than two will receive no credit.

Homework assignments should be submitted either in class or to the envelope on my office door (CST 3-179); you will also need to submit code electronically (details are available on the class web site). Assignments are due by the date and time specified on them: **No late assignments will be accepted.**

**Course Topics**

**Other Information**

**Ethics**
I expect all students to behave ethically: **do not cheat, plagiarize, or commit fraud.** Fraud includes faking program transcripts to make it appear that code works correctly when it does not; plagiarism includes using someone else’s work without proper credit.

If I discover any instances of cheating, fraud, or plagiarism, I will give the guilty parties **failing grades for the course** and report the culprits to the director and the dean. If you are unsure whether a certain action constitutes cheating, fraud, or plagiarism, assume that it does: you may ask me for clarification at any time.

Every student must read and sign a copy of the course **Honor Policy,** which details your obligations to behave ethically. Students will receive zeroes on all assignments/exams until this sheet is turned in to me.

**Accommodations**
Students who may need accommodations because of a disability should make an appointment to meet with the instructor and register with the Office of Student Assistance (306 Steele Hall) and/or the Center for Academic Achievement (804 University Avenue, Room 303).